VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV ELEKTROTECHNOLOGIE
DEPARTMENT OF ELECTRICAL AND ELECTRONIC TECHNOLOGY

NÁVRH ŘÍZENÍ A KONSTRUKCE ROBOTICKÉ RUKY
CONTROLLING AND CONSTRUCTION OF ROBOTIC ARM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **David Sobota**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Vyroubal, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Mikroelektronika a technologie**
Ústav elektrotechnologie

Student: David Sobota *ID:* 154921
Ročník: 3 *Akademický rok:* 2015/16

NÁZEV TÉMATU:

Návrh řízení a konstrukce robotické ruky

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte řízení robotické ruky s šesti stupni volnosti pomocí servo motorů a PC tak, aby ji bylo možné objektově programovat. Návrh konstrukce proveďte v systému SolidWorks. Realizaci dílů pro výrobu proveďte za pomoci 3D tiskárny. Vytvořte demonstrační program.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 8.2.2016 *Termín odevzdání:* 2.6.2016

Vedoucí práce: Ing. Petr Vyroubal, Ph.D.
Konzultant bakalářské práce:

doc. Ing. Jiří Háze, Ph.D., *předseda oborové rady*

Abstrakt:

Práce se zabývá návrhem a vytvořením robotické ruky včetně programu pro desku řízení a PC aplikací pro objektové programování. Řeší, jak vhodně propojit počítač s deskou a jak vytvořit grafické rozhraní pro programování paže. Téma práce vzniklo jako potřeba vytvoření výukové pomůcky pro předměty *Počítačové projektování výrob, logistika a ekologie výroby* a *Technologické projektování a logistika*.

Abstract:

This work deals with creating of robotic arm including program for processor and computer interface with graphical programming for robotic arm. It search for solution, how to connect computer with board fittingly and how to create graphical interface for programming of the arm. Topic of this work is originated as the need to create learning tool for school subjects *Computer projecting of production, logistic and ecology* and *Technological projecting and logistic*.

Klíčová slova:

Robotika, mikrokontrolér, programování, Arduino, servo, řízení, USB rozhraní, 3D tisk.

Keywords:

Robotics, microcontroller, programming, Arduino, servo, control, USB interface, 3D print.

SOBOTA, D. *Návrh řízení a konstrukce robotické ruky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 33 s.Vedoucí bakalářské práce Ing. Petr Vyroubal, Ph.D..

Prohlášení autora o původnosti díla:

Prohlašuji, že jsem tuto vysokoškolskou kvalifikační práci vypracoval samostatně pod vedením vedoucího semestrální práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury. Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení.

V Brně dne 02.06.2016 .....

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Petru Vyroubalovi, Ph.D. za zajímavé téma, za pomoc s přípravou, odbornou konzultaci a poskytnutí zázemí a materiálu pro výrobu komponent.

A dále děkuji doc. Ing. Jiřímu Vaňkovi, Ph.D. za konzultaci k požadavkům na navrhované řízení a aplikaci.

V Brně dne 02.06.2016 .....

Obsah

Úvod

Záměrem práce je vytvoření robotické ruky, naprogramování mikrokontroléru řízení, výroba ovládacího panelu desky a počítačového rozhraní k robotické ruce, jakožto výukové pomůcce do předmětů BPPV (Počítačové projektování výrob, logistika a ekologie výroby) a MTPL (Technologické projektování a logistika). Toto rozhraní bude studentům umožňovat ovládat robotickou ruku přímo anebo jí naprogramovat definované úkony a to vytvořením grafického blokového diagramu, resp. jednoduchým jazykem instrukcí. Program bude i s blokovým diagramem pracovat jako s textovými instrukcemi.[1]

Robotická ruka řešená v práci je vymodelována v návrhovém systému Solidworks a vyrobena technologií 3D tisku. Umožňuje 6 stupňů volnosti a čelist pro uchopení předmětů. Pro pohyb všech mechanismů ruky se využívá servo motorů. Ovládání serv zajišťuje Arduino s přídavnou deskou pro připojení všech serv a ovládacích a informačních prvků. Jako uživatelské rozhraní slouží aplikace, umožňující jak přímé ovládání pomocí posuvníků, tak textové a grafické programování.

Napájení je řešeno externím napájením přes desku řízení a Arduino. Pro spolehlivé ovládání je potřeba vytvořit komplexní řešení zahrnující obvod s mikrokontrolérem a rozhraní pro komunikaci s počítačem včetně aplikace s grafickým programováním. Řešené části řízení tedy jsou:

- Konstrukce robotické ruky – zvažované varianty a výběr jedné pro realizaci
- Vytvoření programu pro řízení Arduina – zabývá se deskou Arduino a nástavbou řízení pomocí přídavné DPS.
- Aplikace a její část odpovědná za komunikaci s mikrokontrolérem – Přibližuje komunikaci USB a virtuální Sériovou komunikaci a úpravu dat pro přenos
- Grafické a textové rozhraní pro ovládání robotické ruky – Rozvádí jak uživatelské možnosti rozhraní, tak technické řešení jednotlivých funkcí.

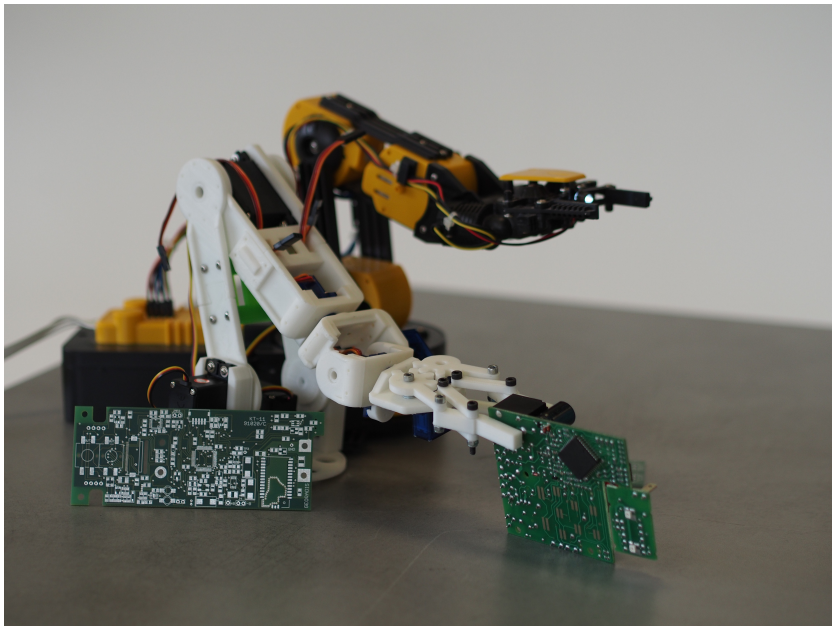
1. Mechanika robotické ruky

1.1. Popis použité robotické ruky

1.1.1. Volba vhodné koncepce

Pro výběr robotické ruky bylo zvažováno více variant. Jako první možnost byla zvolena a zkoumána možnost použití hotového řešení elektro-mechanické stavebnice. Ta byla v jednom kusu již zakoupena a jevila se jako vhodná varianta. Stavebnice však neobsahovala senzoriku nebo jiné zařízení pro ovládání přesné polohy. Pro tento účel bylo navrženo přípravků pro modifikaci kloubů a odečítání jejich polohy. Výhodou této možnosti je méně ovládaných prvků a hotová hlavní konstrukce, nevýhodou jsou úpravy, které se musí na hotovou část namontovat a budou vyčnívat.

Druhou variantou byla konstrukce navržená a vymodelovaná ve vývojovém systému Solidworks. Tato volba dává více možností, jak v počtu kloubů, tak v jejich typu a pořadí (ohyb, otočení). Další výhodou je možnost úprav ještě před výrobou a konečný produkt bude celistvější a nebude potřeba dalších implementací speciálních přípravků. Pro práci byla zvolena tato varianta. Konstrukce vychází z OpenSource práce[2] a je upravena pro použití v bakalářské práci.



Obr. 1 Fotka obou zvažovaných variant

1.1.2. Všeobecný popis

Robotická ruka má 6 stupňů volnosti a čelist pro úchop:

- otáčení ruky do stran přímo v základně
- 3 klouby pohyblivé v horizontální ose (ve výchozí pozici otočných kloubů)
- 2 klouby pohyblivé v rotaci
- čelisti, sevření přes systém pák

Základna je tvořena klecí s otočnou deskou. Ta obsahuje uchycení serv pro první kloub zdvihu ramene. Na konci první části ramene je druhý kloub, servo tohoto kloubu se nachází až na další části. Ta mimo to obsahuje servo pro první rotaci. Mezi oběma rotacemi se nachází kloub posledního ohybu. Poslední část obsahuje servo a systém ozubených kol a pák pro sevření čelisti. Pevné části jsou sešroubovány a pohyblivé spoje využívají serv nebo čepů.

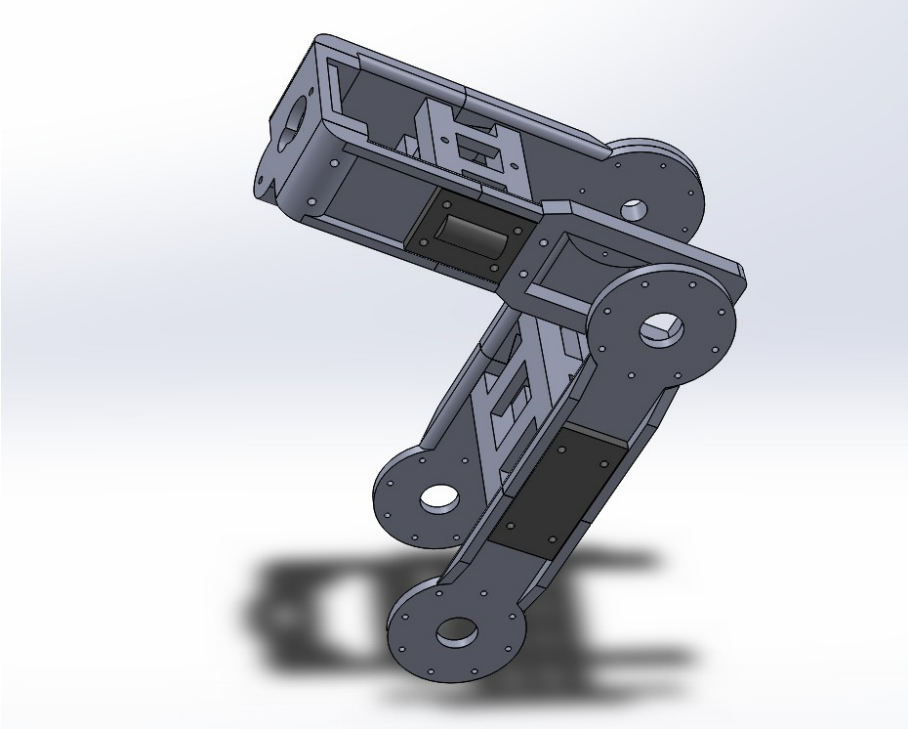
1.1.3. Základna

V základně je použito jedno servo HiTEC HS-311 s rozsahem 180°. Je umístěno vespod základny. Základna je tvořena dvěma kusy, které jsou k sobě spojeny. Má podobu klece s kruhovým půdorysem o průměru 85mm a výšce 50mm. Horní podstava klece je otevřená pro výstup serva. To je vloženo dovnitř a rameno serva je přišroubováno k otočné plošině.

Otočná plošina klec z horní strany uzavírá. V ose otáčení ramene vystupují dvě držadla serv. Jelikož tento kloub je prvním kloubem zdvihu ramene, zdvíhá nejvyšší zátěž a proto obsahuje dvě serva, na každé straně kloubu jedno. To je také důvod, proč není potřeba pro osovou souměrnost použití čepů.

1.1.4. Paže

Vlastní paže začíná právě kloubem se dvěma servy HS-311. První část má délku 145mm a skládá se z několika dílů, především pak dvou totožných dvoudílných ramen. Oba díly každého ramene mají zámky a otvory na šrouby pro pevný spoj. Jako spojka zde slouží díl, který zároveň slouží jako vzpěra. Do ní jsou šrouby spoje vedeny a mimo rozdělení zámků zabraňuje také kroucení tohoto ramene. Toto rameno neobsahuje žádné připevněné servo a pohyb obou jeho konců je zajištěn servy z navazujících částí.

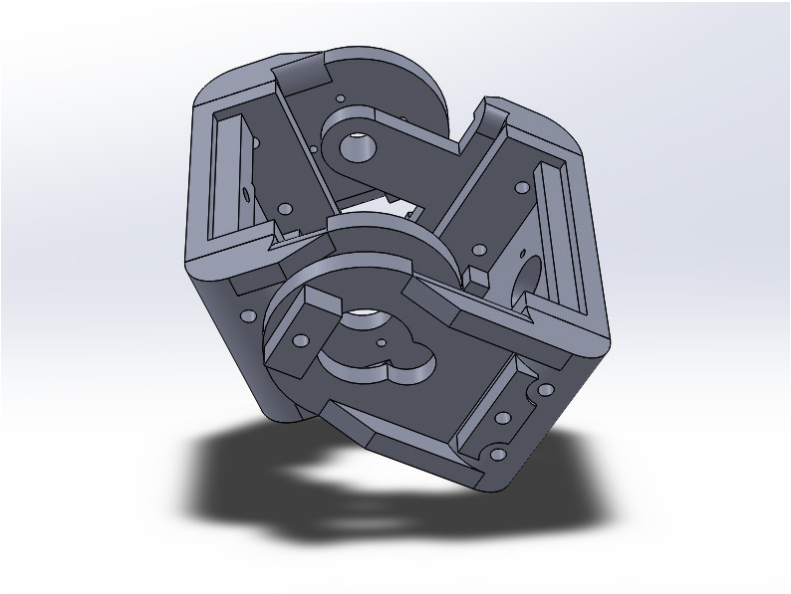


Obr. 2 Model dvou částí paže bez osazených serv

Druhá část ramene obsahuje serva na obou koncích. Kloub společný s prvním ramenem pohání opět servo HS-311. Toto rameno je tvořeno čtyřmi díly. Jeden díl obsahuje slot pro servo a jeho protikus obsahuje čep. Oba tyto díly jsou spojeny jednak vzpěrou podobnou, ale především koncovým dílem, obsahující druhé servo. Jedná se o servo GO-09 s pracovním úhlem 180°. Toto servo zajišťuje první rotaci na paži mimo rotace s celou paží v základně.

1.1.5. Kloub rotací a ohybu

Tato část ruky slouží jako předloktí a zápěstí. Zajišťuje totiž rotaci a tím možnost otočení uchopeného předmětu. Skládá se ze šesti dílů, jež jsou pevně spojeny do dvou celků, které jsou schopny se vůči sobě ohýbat. Rotaci vůči druhému ramenu paže zajišťuje druhé servo paže a to v rozsahu 180°. V ose na druhém konci celku se nachází druhé servo GO-09 umožňující rotaci. To dává možnost rotovat až o 360° a uchopené předměty otáčet hlavou vzhůru.

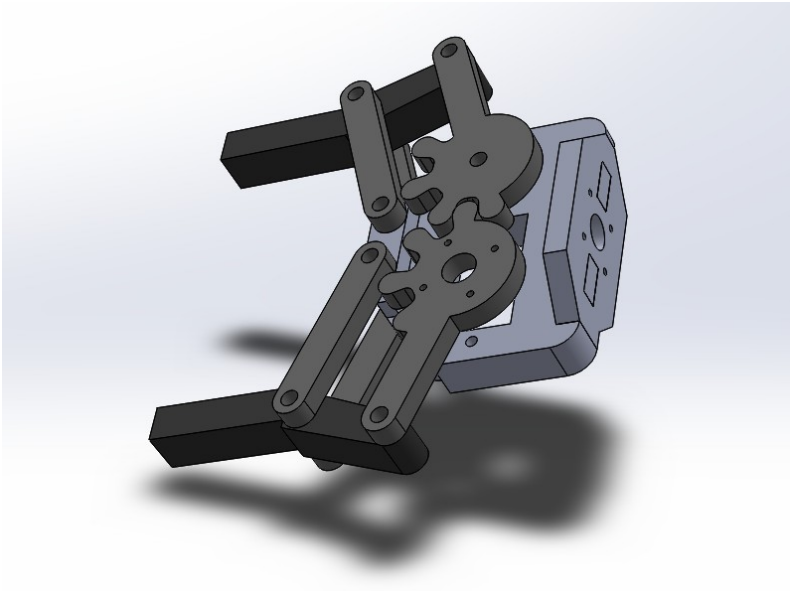


Obr. 3 Celek zajišťující 2 rotace a ohnutí

Napříč celkem vede pomyslná osa ohybu, jíž zajišťuje na jedné straně čep a na druhé servo GO-09 jako v předchozích případech. Při vhodné konfiguraci pohybů je možné toto uskupení kloubů využít pro otáčení předmětů vzhůru nohami, otáčet je ve vertikální ose o 180° nebo pokládat předměty na velmi rozdílné výškové hladiny v kolmé poloze, což je další výhodou tohoto řešení konstrukce.

1.1.6. Čelisti

Na servo z předchozího celku navazuje tělo čelisti. To je možné celé rotovat a zespodu (ve výchozí poloze serva rotace) je připevněné servo GO-09 pro pohyb čelisti. K tělu jsou připevněna dvě ozubená kola s pákami. Jedno je poháněno servem a zajišťuje pohyb levé části čelisti. Druhé využívá k pohybu ozubení a pohání přes systém pák pravou část čelisti. Dále jsou k tělu připevněna ještě dvě ramena na každé straně, která zajišťují rovnoběžnou polohu ploch čelistí vůči sobě i vůči tělu čelisti. Koncová část čelisti, sloužící k uchycení předmětu, je zalomené rameno připevněné k oběma ramenům. Spojení je provedeno šrouby, které slouží jako čepy.



Obr. 4 Model čelisti v detailu

1.2. Ovládání pohybu

Pro ovládání přes počítač nebo mikrokontroler je potřeba znát aktuální polohu ramene nebo minimálně výchozí polohu. Při výběru vhodného řešení konstrukce bylo zvažováno více variant a to včetně ovládání pohybu. Níže jsou uvedeny a popsány všechny metody, jak ovládání zrealizovat.

1.2.1. Metody koncových poloh

První řešení, nechat motor zinicializovat (dojet na koncovou polohu) nepřipadá v úvahu. V takovém případě je potřeba nechat jet všechny motory dost dlouho, aby stihly dojet na koncovou polohu i v případě, že se nacházejí na zcela opačné koncové poloze. Avšak v případě, že se nachází na začátku inicializace blízko požadované koncové polohy, je motor a ozubená soukolí i přes ochranná opatření zbytečně zatěžovány. Ale ani po inicializaci nebude určování polohy přesné. Pro její určení se bude pracovat pouze s dobou napájení pohonu daného kloubu a definovanou rychlostí pohybu kloubu. Problém může nastat při kolísání napětí nebo zátěži. A k zátěži se dá započítat i vlastní hmotnost konstrukce. To by znamenalo jinou rychlost u klesání než u stoupání a navíc jinou rychlost závislou na hmotnosti zdvíhaného břemene. Tato metoda je proto nevhodná.

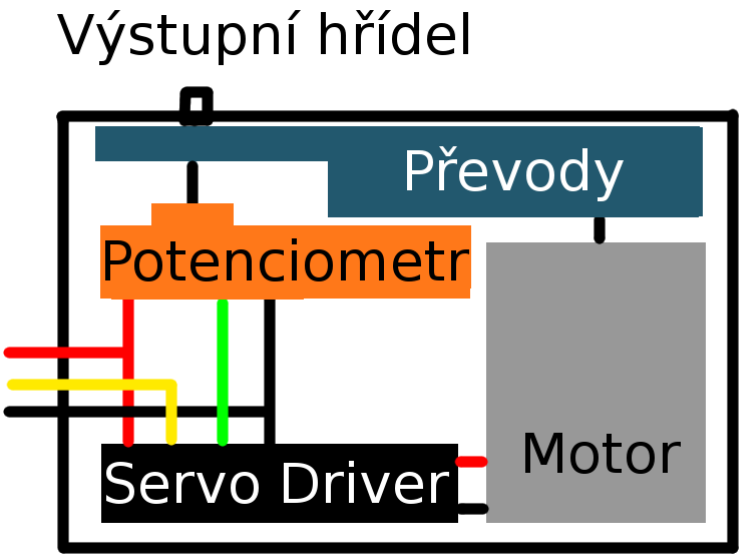
Druhým řešením je osazení senzorů koncové polohy. Ty budou zajišťovat přerušení dodávky proudu v případě dosažení konce chodu ramene. Stejně jako u předchozího řešení bude potřeba použít inicializace, avšak v tomto případě se může využít signál z koncových poloh například pro zkrácení inicializace v případě, že se rameno nachází v požadované poloze. Ale ani toto, stejně jako předchozí, nezajišťuje kontrolu polohy ramene kromě koncových poloh. Poloha je řízena opět pouze délkou času pohybu a zpětná vazba chybí.[1]

1.2.2. Metoda přesných poloh – kontrola pomocí optické závory

Metody přesných poloh jsou již schopny řešit i pohyb o nekonstantní rychlosti, jelikož sledují nebo řídí přesnou polohu kloubu. Jednou z těchto metod je sledování pohybu. Spočívá v montáži optické závory do kloubu. Některé z ozubených kol v kloubu by bylo potřeba dokola navrtat dírami, aby optická závora byla schopna zaznamenávat rotaci motoru a soukolí. V kloubu je více převodů, takže by se frekvence výstupního signálu optické závory lišila podle sledovaného ozubeného kola. Tato úprava však vyžaduje zásah do pohonu a oproti dalším řešením je potřeba další komunikace s mikrokontrolérem a využití jeho výpočetní kapacity pro počítání polohy. Navíc se tímto nevyřešil zjištění polohy po zapnutí a opět je nutná inicializace koncové pozice kvůli počítání impulzů od této koncové polohy.[1]

1.2.3. Metoda přesných poloh – řízení servo motory

Posledním řešením, při porovnání nejvhodnější, je použití servo motorů. Na rozdíl od předchozího případu s fotozávorou není poloha kontrolována mikrokontrolérem řízení a není potřeba inicializace koncovou polohou. To je možné díky principu servo motoru. Ten je složen ze čtyř klíčových částí. Již z jeho názvu je jasné, že obsahuje motor. Jedná se o standardní DC motor, nikoli krokový. Pro zjemnění chodu a zvýšení síly jsou použity převody. Na výstupu ozubeného soukolí se nachází potenciometr, který určuje svým rozsahem také rozsah otáčení serva. Navíc dle výchylky serva a tím své, mění poměr odporu/napětí a tím nastavuje řídící klopný obvod. Ten je poslední částí serva a spolu s potenciometrem tvoří servo ovladač (servo driver).



Obr. 5 Vnitřní blokové schéma servo motoru [3]

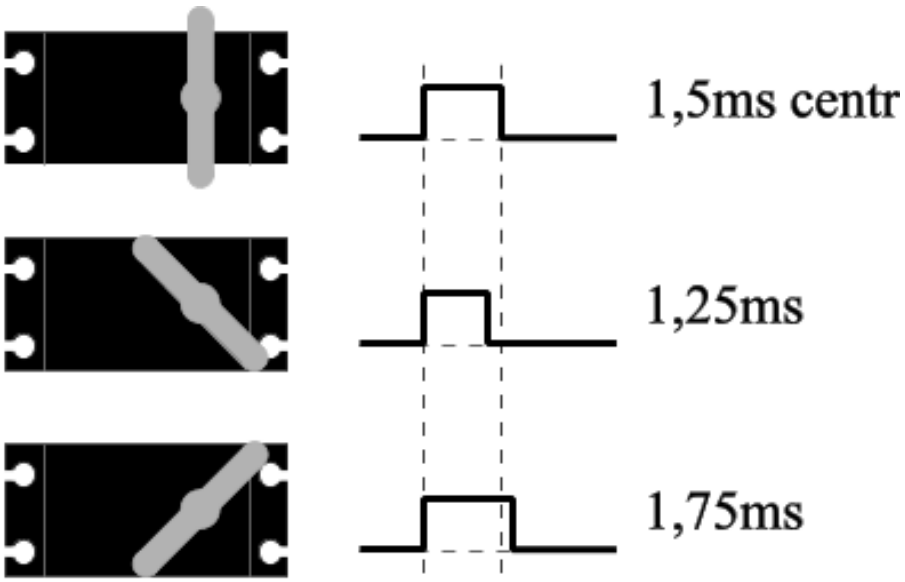
1.2.4. Řízení serva přes PWM

Servo je z vnější řízeno impulsy, jejichž střída (proměnná šířka pulsu při konstantní periodě) určuje polohu. Řídící elektronika serva si na základě odečteného napětí z potenciometru také generuje vlastní impulsy o odpovídající střídě. Signál přijatý z vnějšího řízení následně porovnává se signálem generovaným podle napětí. Střída vnitřního generovaného signálu se přibližuje střídě signálu řídícímu tím, jak se mechanizmus serva otáčí k požadované poloze. Směr otáčení se také určuje porovnáním stříd signálů, konkrétně jejich rozdílem. Po dosažení cílové polohy je nastolena rovnováha signálů a jejich rozdíl je roven nule. V ten okamžik elektronika přestane dodávat motoru energii.[4]

Tyto vlastnosti předurčily servo jako nejvhodnější řešení pro ovládání mechaniky robotické ruky. Pro původní konstrukční řešení by bylo potřeba úpravy právě pohonu. Volba serva a konstrukce pro něj přizpůsobené je nejvhodnější volbou.

Parametry PWM řídících signálů serv se liší dle výrobce a typu. V současné době se nejčastěji využívá k řízení pulsů o délce v rozmezí 1 – 2 ms. Opakovací frekvence pulsů je většinou okolo 50 Hz, ale opět se může typ od typu lišit. Frekvence opakování řídícího pulsu pouze změní reakční dobu serva (jak dlouho trvá, než se rozpohybuje). Tento parametr však nemůže způsobit nefunkčnost serva, pokud je v rámce rozumných mezí a nemění se v průběhu času.

Na obrázku níže je možné vidět 3 případy řídícího signálu serva. Jedná se o servo s pracovním úhlem 180° a krajní hodnoty pulsů jsou 1 ms pro nejkratší a 2 ms pro nejdelší. Přesně uprostřed tohoto rozsahu, při 1,5ms pulsu, je výchylka serva právě 90°. Hodnoty 1,25 ms a 1,75 ms odpovídají čtvrtinám maximálního rozsahu (45° a 135°). Přesnost výchylky je určena přesností pulsů. Proto jsou serva připojena na PWM výstupy mikrokontroléru, které jsou pro toto použití optimalizovány. [4]



Obr. 6 Šířky impulsů a odpovídající výchylky serva [5]

1.2.5. Použitá serva

V konstrukci robotické ruky jsou použity dva typy serv. Větší silnější serva jsou umístěny ve spodních částech zařízení. Zde je potřeba pohybovat větší hmotností (hmotnost předmětu + většiny paže) a jejich vyšší hmotnost a velikost nebudou tolik překážet. Slabší a lehčí serva se nacházejí výše na ruce, kde není potřeba takových sil a naopak je nutností nízké hmotnosti.

Jako těžší serva jsou použity Hitec HS-311 Standard. Je napájeno stejnosměrným napětím 5V. Použitelný rozsah otáčení je 180° (omezen možnostmi mikrokontrolérem). Tělo i převody jsou plastové. Tah serva při 4,8V je 3 $kg.m^{-1}$.

Obr. 7 Větší servo Hitec HS-311 [6]

Lehčí a menší servo GO-09 je také napájeno stejnosměrným napětím 5V a má stejný rozsah pohybu. Převody i tělo jsou plastové. Tah serva při 4,8V je 1,3 $kg.m^{-1}$.

Obr. 8 Mikroservo GO-09 použité na ramenu[7]

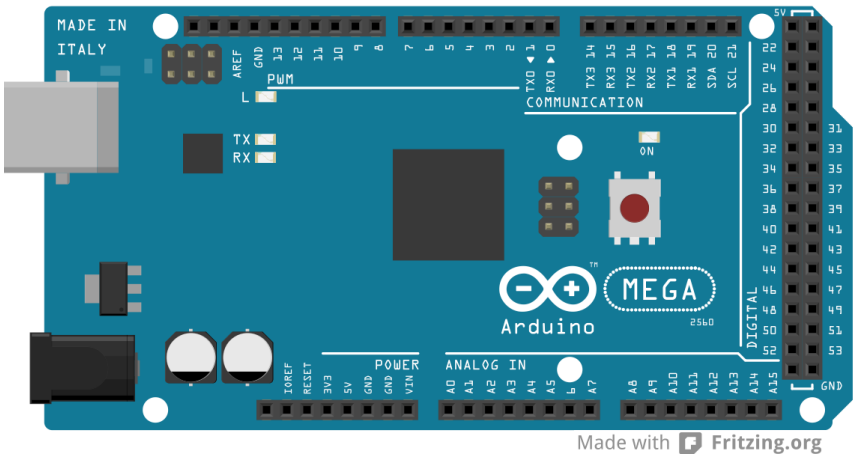
2. Řízení s mikrokontrolérem

2.1. Arduino

2.1.1. Základní parametry

Jako řízení bude použita deska Arduino Mega. Její délka je 53,30 mm a šířka 101,52 mm. Obsahuje mikrokontrolér Atmega 2560 architektury AVR. Napájení čipu je 5V a doporučené napájení desky je 7-12V. Arduino obsahuji stabilizátor 5V přímo na desce. Ideální je možnost napájení všeho přes USB konektor, Jelikož by však proudový odběr přímo z USB byl příliš velký, Arduino bude napájeno, stejně jako pohon vlastní robotické ruky, přes DPS řízení z externího adaptéru 9V nebo 12V. Napájení bude jednotné pro řízení i mechaniku a tím také praktické pro použití v laboratoři. Případné stabilizátory pro různá napětí budou umístěny na desce řízení.[8]

Arduino Mega dále obsahuje 54 digitálních vstupů/výstupů (15 z nich umožňují PWM výstup) a 16 analogových vstupů. Přímo na desce je 16MHz krystalový oscilátor. Dále obsahuje více typů pamětí: EEPROM s kapacitou 4KB, SRAM s 8KB a Flash paměť o celkové kapacitě 256KB, z nichž 8KB je využito pro zavaděč.



Obr. 9 Deska Arduina Mega 2560 [9]

2.1.2. Využití pro projekt

Jako řídící výstupy budou použity PWM výstupy, které jsou pro tuto aplikaci vhodné, jelikož řídící signál serva PWM využívá. Komunikace s počítačem bude probíhat přes rozhraní USB, které slouží také k programování. Ke komunikaci však nebude využito programování mikrokontroléru, ale komponenta ve vývojovém prostředí pro jazyk C++ aplikace a vytvořený vyšší jazyk pro ovládání robotické ruky. Zbylé piny desky budou využity pro informační display nebo jiné informační a ovládací prvky. Pro tento účel bude vytvořena deska plošných spojů zapojena do dutinkové lišty Arduina.

2.2. DPS řízení

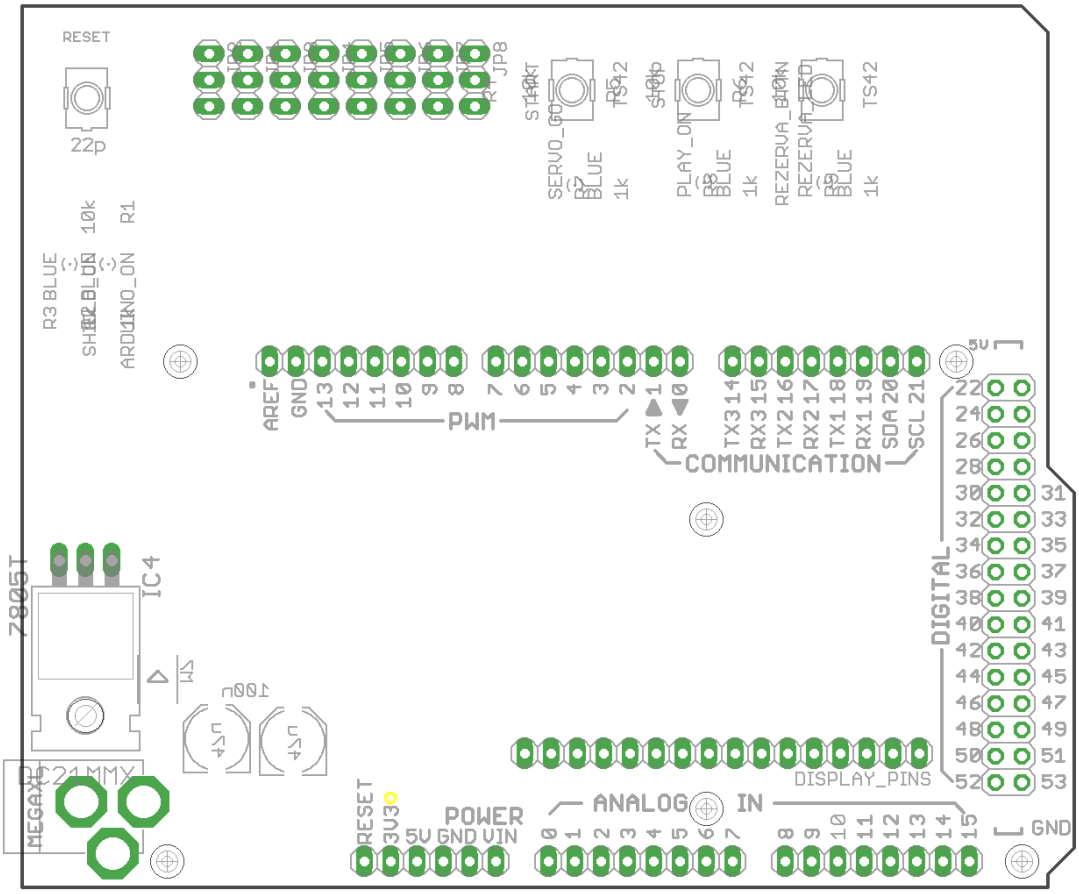
Pro lepší konektivitu řízení k robotické ruce bude vytvořena deska plošných spojů, která bude přesně pasovat na dutinkovou lištu Arduina. V podstatě se bude jednat o shield (rozšíření Arduina) vyrobený na míru. Pro rozměr desky a rozmístění konektorů bude využita open source dokumentace k Arduinu. Mimo konektivitu bude deska zajišťovat ovládací a informační rozhraní.

Jako výchozí šablona pro přesné rozmístění kolíkových lišt poslouží schéma ve složce příklady v Eaglu. Zespodu budou po obou stranách kolíky dle šablony Arduina. Z vrchní části pak budou umístěny konektory pro ovládání motorů a konektor pro externí napájecí adaptér. Z vrchní části budou také umístěny ovládací a informační prvky, aby byli dobře přístupné. Deska obsahuje svůj vlastní stabilizátor kvůli odběru serv. Celek je napájen vhodným adaptérem (9-12V).Ve vrchní části je možné připojit informační LCD displej. Pro lepší čitelnost bude podsvícen. Je možné použití i LED varianty pro lepší kontrast znaků nebo jinou barvu. Displej je možné připojit do dutinkové lišty, takže záměna za jinou barvu nebo LCD za LED bude jen otázka pořadí pinů a případná malá změna v programu.

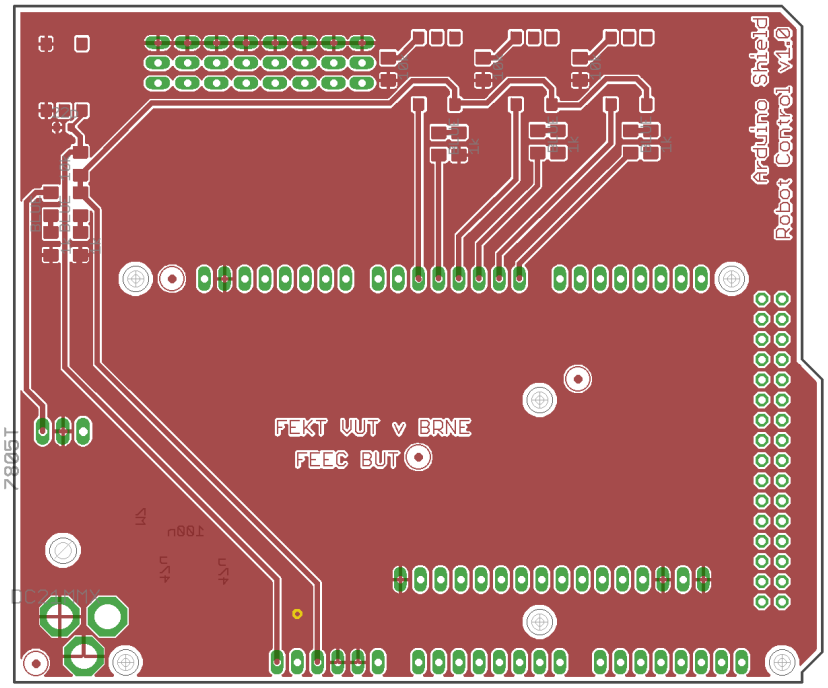
Kromě displeje jsou na desce i další informační prvky a to LED diody. Ty budou signalizovat především stav napájení. Napájecí indikátory jsou dvě modré LED. Jedna kopíruje stav kontrolky přímo z Arduina. Na desce bude z důvody zakrytí originální diody na Arduinu přídavnou deskou. Druhá dioda indikující napájení je zapojená za stabilizátor 5V a bude podávat informaci o napájení serv. Dále jsou tu ještě 3 svítivé diody, indikující právě vykonávaný pohyb serva (práci Arduina) a spuštěný nebo vypnutý režim přehrávání uložené sekvence. Třetí LED je na desce jako rezerva pro případné rozšíření funkcí obvodu.

Deska obsahuje také ovládací prvky a to 4 tlačítka. První kopíruje funkci resetu Arduina, který je vyveden přímo z desky mikrokontroléru. Toto tlačítko je tedy jednoúčelové. Zbylé tři tlačítka jsou připojeny na programovatelné digitální vstupy/výstupy. Jedno slouží jako spouštěč uložené sekvence a druhé slouží k ukončení této sekvence. Třetí tlačítko je opět jako rezervní.

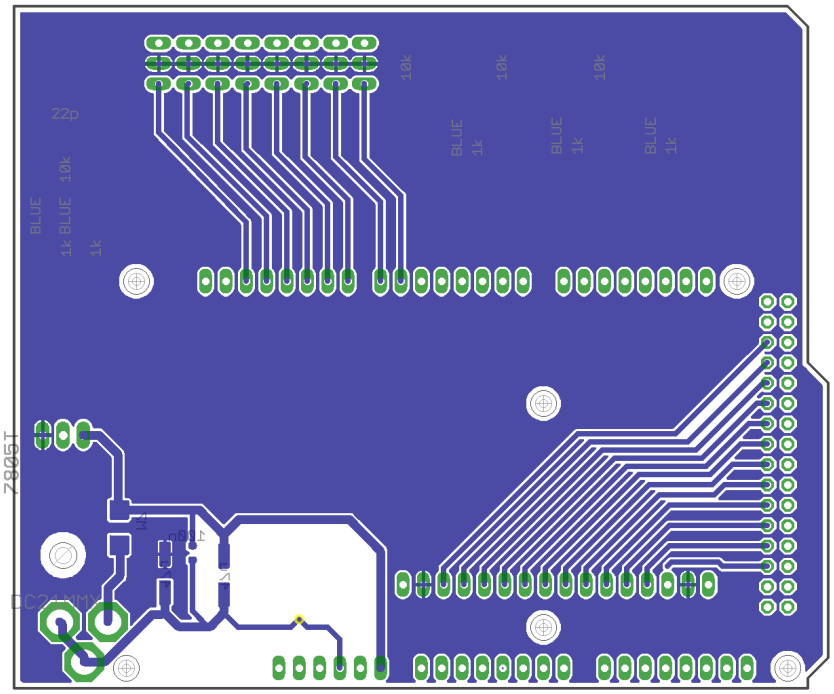
Deska je kvůli displeji větší než standardní Arduino Mega a na jedné straně ho přesahuje. Obvod je řešen na dvoustranné desce a obsahuje převážně pinové nebo dutinkové lišty.



Obr. 10 Výkres desky se součástkami



Obr. 11 Výkres desky ze strany součástek



Obr. 12 Výkres spodní strany desky

2.3. Program mikrokontroléru

Mikrokontrolér bude obsahovat program, který se dá rozdělit z funkčního hlediska na několik hlavních částí.

2.3.1. USB/Serial komunikace

První část, z hlediska komunikace, bude komunikace přes USB. To je sice využito fyzicky, ale virtuálně se mikrokontrolér připojuje na sériový port. Je to dáno vlastností Arduina, které pro komunikaci s vývojovým prostředím Arduino IDE používá, mimo programování, také pouze sériový port. USB je používáno jen pro nahrávání firmwaru mikrokontroléru a o tuto operaci se stará čip Atmega 16U2, který není vlastnímu mikrokontroléru dostupný. Arduino se k počítači připojí jako zařízení Plug and Play a bude čekat na další instrukce. Těmi je přepis firmware přes USB a nebo připojení jako zařízení na sériovém portu. Režii USB komunikace provádí automaticky systémové ovladače na pozadí. O režii komunikace sériového portu se stará aplikace s vývojovým prostředí. Komunikace probíhá dvousměrně přes signály Rx (příjem) a Tx (odeslání). Obě zařízení komunikace (mikrokontroler i počítač) čekají na pozadí na příjem dat a provádí jiné své operace. Při příjmu dat do svého přijímacího zásobníku tuto událost zaznamenají, dokončí aktuální činnost a poté data vyčtou a dále vyhodnotí, uloží nebo provedou.

V rámci USB/Sériového rozhraní bude probíhat i další komunikace. Firmware pro tuto aplikaci má v sobě unikátní podpis. Při otevírání portu počítačovým rozhraním je poslán do mikrokontroléru povel pro zahájení připojovací sekvence. I když je Arduino k počítači již připojeno přes USB, pro sériový port není spojení stále vytvořeno. Na zahajovací povel *connect* reaguje Arduino odesláním série řetězců s informacemi o sobě a firmwaru. Nejdříve se v aplikaci zobrazí jako kontrola reálného připojení hláška *CONNECTED.. CHECKING FIRMWARE..* která není pro počítač důležitá a má spíše informativní charakter pro uživatele. Dále odešle název svého firmwaru. Vzhledem k více druhům prostředků Arduino je v první části uveden název desky, v našem případě Arduino Mega 2560. Za pomlčkou následuje název programu, většinou obsahující funkci, tedy Robot Control App. Celý název je načten do stavového řádku, zobrazen v informačním panelu a po porovnání je zobrazeno případné upozornění, že firmware nemusí být zcela kompatibilní s počítačovou aplikací nebo s robotickou rukou. Pro zjištění kompatibility je klíčová také verze nahraného programu. Pro starší aplikaci se může novější deska jevit jako téměř zcela neznámá, v novější aplikaci bude přidána databáze všech předchozích verzí firmwarů desek.

2.3.2. Ovládání ruky

Nejdůležitější část programu, ovládání robotické ruky, má za úkol volání čtení instrukcí z paměti, převod instrukčních slov na řídící signály a distribuci těchto signálů na výstupy Arduina. I když bude jakýmsi jádrem programu, bude se podřizovat řízení z komunikace USB a nebude se moci spustit při zápisu paměti. Před online režimem však bude mít přednost. Plnou kontrolu nad mikrokontrolérem bude mít v případě odpojení od počítače, kdy se stane nezávislým systémem. Kromě hlavních funkcí bude instrukce posílat informační části programu a také bude přijímat některé řídící signály z tlačítek desky řízení (start, stop, init, reset a jiné).

V okamžik, kdy dostane od USB nebo z desky řízení povel pro spuštění, začne z paměti vyčítat postupně instrukční slova. Podle prvního písmena, tj. velkého písmene určující příkaz rozhodne, pro který kloub (který výstupní pin) je instrukce určena. V prvním písmenu je také obsažena, zda se bude měnit rychlost serva a nebo zda bude následovat prodleva/pauza. Po rozhodnutí vezme druhou slabiku slova a použije ji jako číselnou hodnotu pro délku (střídu) pulsu, popř. Interval změny polohy nebo délku pauzy.

V jeden okamžik je v pohybu vždy jen jedno servo, což zjednodušuje uživateli programování a sestavování sekvence pohybů. Při použití online režimu je Arduino schopné posílat krátké zprávy po dosažení polohy. Možnost nezávislého programování a současné možnosti ovládání ruky v online režimu umožňuje uživateli během programování vyzkoušet v online režimu konkrétní polohu a pak ji zadat do kódu.

I když je možné pohyb řídit velmi přesně, řízení nemá reálné informace

2.3.3. Informační výstupy

Poslední částí bude informování uživatele o tom, co mikrokontrolér právě dělá skrze desku rozhraní. Data z ostatních částí programu budou převáděna do textu a posílána do LCD displeje ve vhodných intervalech. Na displeji tak uživatel bude moci vidět hlášení jako PROGRAMMING (programování), MANUAL (ruční řízení), AUTO (přehrávání instrukcí z paměti), INIT (inicializace) nebo přímo aktuálně prováděná operace robotické ruky ZAKLADNA 30 (otočení základy do polohy 30°) nebo WAITMS 500 (čekání 500 ms). Dále zde budou informační LED, které budou určité stavy a informace zobrazovat zvlášť. Jako třeba napájení, stavy *manual* nebo *auto*, *play* nebo *stop* a také pohyb aktuálního serva (LED připojené na výstupy serv). Nebude tu chybět ani indikace chybového stavu FAULT LED. Pro lepší přehlednost je možné použití dvoubarevných LED diod pro signalizaci pohybu a dosažení polohy kloubů.

3. PC aplikace

Počítačová aplikace je realizována pomocí programovacího jazyku C++, konkrétně formulářová aplikace CLR využívající .NET4 Framework, ve vývojovém prostředí Visual Studio 2010. Funkčně se pak celá aplikace dá dělit na komunikaci s řízením přes USB, která bude probíhat na pozadí, a uživatelské rozhraní, ve kterém bude možné ruku ovládat přímo anebo nadefinovat sekvenci úkonů. Mimo spustitelného souboru .exe využívá aplikace také souborů v adresáři *AppData*. Zde je uložen například soubor pro ukládání posledních souborů nebo externě načítané obrázky, které nebylo možné implementovat přímo do spustitelného souboru.

3.1. Komunikace s řízením

Všechny instrukce pro mikrokontrolér budou v programu přeloženy na signály kompatibilní s programem mikrokontroléru. USB (Universal Serial Bus) patří mezi nejrozšířenější způsob připojení k počítači. Veškerá komunikace je v režii zařízení Master (počítač) a data od mikrokontroléru si program musí vyžádat. Pro přenos jsou vymezené rámce o časové délce 1 ms. Vlastní data se pak posílají v paketech o minimální délce 8 bajtů. Délka paketů se může pohybovat v od minima až po 256 bajtů. Jeden rámec může obsahovat pakety i pro více zařízení.

Jednotlivé bity jsou na lince kódovány metodou NRZI (Non Return to Zero Inverted), logická 0 tedy vede ke změně úrovně signálu na lince, logická 1 úroveň ponechává. O kódování a dekódování signálu linky se stará hardware bez vlivu na software. Kromě této metody se používá vkládání bitů pro vynucení změny úrovně. Konkrétně po šesti výskytech logické 1. Každý datový paket začíná kvůli synchronizaci zaváděcím bytem 0b00000001. USB využívá více virtuálních linek, tzv. trubic (pipes), reálně se jedná o pakety pro různé koncové adresy. Pro zvýšení rychlosti přenosu dat je možné využít více takových trubic jedním zařízením.[10]

3.1.1. Režie sériové komunikace

Arduino však pro komunikaci využívá sériový port, i když v případě USB propojení se jedná o COM port virtuální. Pro komunikaci se sériovými porty je ve Visual Studiu dostupná komponenta *SerialPort*, které je možné přednakonfigurovat základní parametry, jako je rychlost přenosu *(BaudRate)*, počet bitů, parita a řídící signály.

Jedním z důležitých signálů je DTR *(DtrEnable)* – Data Terminal Ready. Jeho defaultním zakázáním se zabrání resetování Arduina při připojení kabelu. To je praktické především proto, že mikrokontrolér bude ovládat mechanické prvky a jejich nekontrolovaný pohyb během resetu by mohl způsobit kolizi a poškození. Vzhledem k použití robotické ruky v laboratořích toto riziko stoupá při neopatrných pokusech o nápravu chyby odpojováním a následným zapojováním konektorů. Arduino je proto možné za běhu připojit bez následků, třeba v rámci troubleshootingu připojení. Řídící signál je však v některých případech potřebný, především když připojené zařízení přejde z nějakého důvodu do neaktivního stavu a odmítá komunikovat. Proto je v aplikaci obsaženo tlačítko inicializace spojení, které parametr *DtrEnable* nastaví na krátkou dobu do stavu True.

Dále je pro korektní připojení potřeba zvolit port. Nástrojová lišta *Připojení* obsahuje tlačítko *Aktualizovat*, které naskenuje a zobrazí všechny aktivní porty do rolovací nabídky *Porty*. Po zvolení portu se zobrazí číslo portu také ve stavovém řádku a poté stačí port otevřít.

Následující kód ukazuje otevírání portu:

```
this->mainSerialPort->Open();
this->mainSerialPort->DiscardInBuffer();
this->mainSerialPort->Write("connect");
_sleep(100);
this->ConsoleOutput->Items->Insert(0,(this->mainSerialPort-
>ReadLine()));
String^ FirmWare = this->mainSerialPort->ReadLine();
String^ Compatible = this->mainSerialPort->ReadLine();
this->ConsoleOutput->Items->Insert(0,FirmWare);
```

V ukázce dochází k otevření portu, po přístupu je mazán příchozí buffer, aby nedošlo ke čtení špatných dat. Poté program pošle příkaz *connect* a čeká 100 ms na odezvu mikrokontroléru. Po přijetí prvních dat je zobrazí v informačním panelu potvrzení desky o připojení. Dále do proměnných, jednotlivých polí ve stavovém řádku a také do informačního panelu zapisuje přijímané data o desce, firmwaru a jeho verzi.

Pod režii komunikace patří také funkce detekující příchozí data, příznaky povolující čtení a zápis dat nebo signalizace připojení pro uživatele. V poslední řadě také ošetření výjimek a chyb, jako je připojování k již nedostupnému portu nebo zápis do zařízení, které se již odpojilo.[11]

3.1.2. Datový proud s kódem

Program při kompilaci převádí každý příkaz na dva znaky ASCII. První popisuje příkaz, zda se jedná o nastavení rychlosti nebo servo a o které. Druhý nese informaci o hodnotě (hodnota v závorce za příkazem). Při delší sekvenci příkazů vznikne souvislý řetězec písmen, který je po dokončení odeslán jako nepřerušený proud do Arduina, kde je celý řetězec uložen do EEPROM. Aby Arduino poznalo, že má řetězec uložit a také aby bylo zajištěno regulérní přenesení ukončení řetezce, je ohraničen ostrými závorkami. Závorka vlevo dává programu povel, aby začal řetězec zpracovávat pro uložení. Závorka vpravo posouvá umístnění ukončení řetězce, které je při posunu vkládáno a zajišťuje, aby nebyl poslední znak řetězce ztracen nebo poškozen.

Kontrola správnosti dat je částečně možná zpětně v mikrokontroléru, kdy se generovaný řetězec řídí pravidlem, že první znak, popisující příkaz, je vždy velké písmeno. Druhý znak nabývá hodnot malých písmen, poté číslic a při nedostačující škále jsou použity speciální znaky *@, &* (použito @ pro poslední hodnotu výchylky serva). Pří čtení z paměti je pak možné kontrolovat, zda je první písmeno velké písmeno a pokud není, bude toto písmeno přeskočeno a bude se číst až další velké písmeno v řadě.

Výsledný řetězec, který je posílán přes sériový port:

```
<AbBsCpDxEsFiGsCaAaAsBs>
```

Mimo posílání řetězce zde jsou ještě povely pro spuštění a zastavení uloženého pohybu nebo případně vytvoření smyčky a také je zde zpětná vazba. Povely jsou posílány jako textové řetězce. Povely stejného typu slouží k online ovládání nebo pro navázání komunikace. Při navázání komunikace se posílá povel *connect*, pro ovládání konkrétního serva povel *SERVO130* (oproti příkazům při programování se jedná o zjednodušený tvar bez závorek, který je procesorem lépe zpracovatelný). Pro spuštění a ukončení smyčky jsou vyhrazeny povely *start* a *stop*.

Z důvodu režie zápisu a čtení paměti bude potřeba určitá režie, která bude ošetřena přímo v aplikaci. V případě vykonávání instrukcí mikrokontrolérem (čtení instrukcí z paměti) nebude možné vyčítat data do počítače nebo nahrávat nové instrukce do mikrokontroléru. To bude zajišťoval zakázání ovládání v aplikaci a její povolení bude možné až po vyslání povelu *stop*. Podobné opatření bude použité i při nahrávání programu do mikrokontroléru. V každém případě změní některé prvky v aplikaci, které by mohli daný proces negativně ovlivnit, stav na Disabled nebo se nastaví příznak, který na pozadí způsobí dočasnou nefunkčnost prvku.

3.1.3. Kompilace a kontrola kódu

Z technických důvodů je kompilace prováděna jen v textovém režimu. Navíc se jedná o výhodnou možnost, jelikož mimo kompilace je i otevírání a ukládání textových dat mnohem praktičtější a méně náročné na výkon. Pro možnost programování a kontroly bez připojení mikrokontroléru je přidáno zvlášť tlačítko pro kontrolu bez kompilování a posílání dat. Obě funkce, jak kontrolní, tak kompilační volají stejné funkce pro kontrolu, jen kompilační funkce volá navíc funkci pro převod na řetězec.

Při kontrole je v informačním panelu zobrazován průběh včetně nalezených chyb a řádku, na kterém se chyba nachází. Píše se zde i typ chyby a možný důvod, proč se chyba zobrazila. Na konci je zpráva, zda byla kontrola úspěšná nebo zda obsahovala chyby a kolik jich obsahovala. Zde je ukázka hlášení chyby špatně zadané hodnoty:

```
if((CheckingSpeedValue(ToCheckValue) == 1)
&&(NotCode == 0)&&(ToCheckCommand=="SPEED_"))
{
  this->ConsoleOutput->Items->Insert(0,
  ">>> Zvolte hodnoty v rozsahu 0 - 20");

  this->ConsoleOutput->Items->Insert(0,
  "CHYBA na řádku [" + LineIndex + "] - chyba rozsahu.");

  ErrorCode = 1;
  ErrorCount++;
}
```

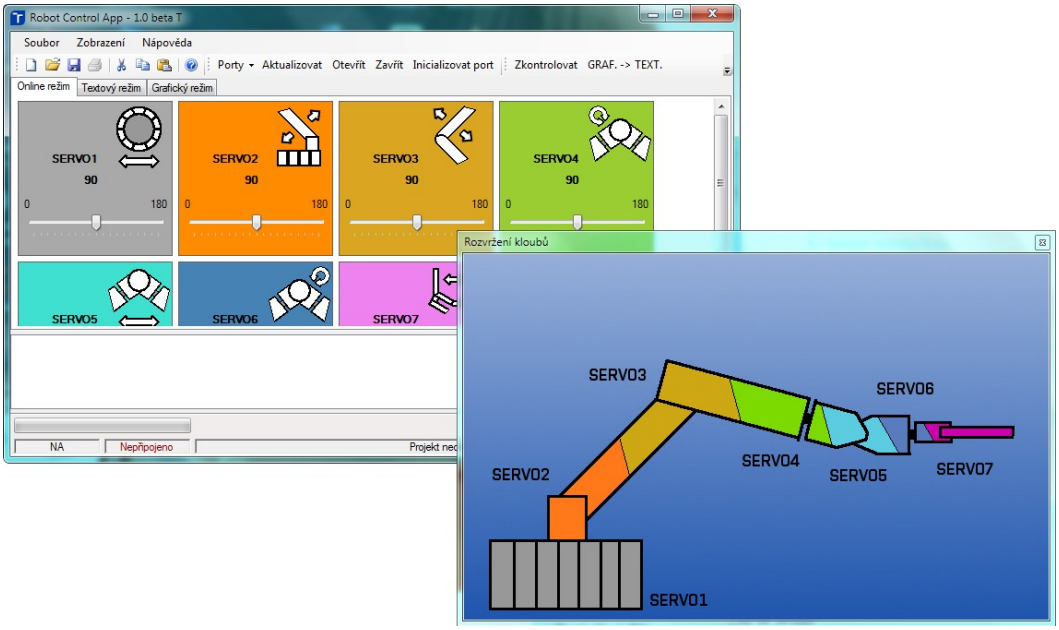
V podmínce se testuje, zda funkce *CheckingSpeedValue* (funkce pro kontrolu správného rozsahu hodnot) vrátila 1 nebo 0 (1 je chyba, 0 je bez chyby) a zároveň zda příkaz má hodnotu *SPEED_*. Tento úryvek kódu vyhodnocuje číselný rozsah hodnoty pouze u příkazu *SPEED_*, jelikož ostatní příkazy mají hodnoty odlišné. Dále je zde příznak *NotCode* roven nule, což označuje, že se nejedná o komentář, ale kód. Pokud jsou pak všechny podmínky splněny (chyba kódu, jedná se o kód a příkaz je *SPEED_*), vypíše se do informačního panelu na první řádek *CHYBA na řádku X - chyba rozsahu.* a na druhý řádek nápověda *>>> Zvolte hodnoty v rozsahu 0 – 20* a přičte se chyba do počítadla.

3.2. Uživatelské rozhraní

Uživatelské rozhraní bude umožňovat více možností ovládání. Grafické a textové online ovládání ruky bez možnosti uložení odvysílaných dat a bez možnosti vytvoření sekvence příkazů. A poté grafické a textové programování, kde není možná okamžitá interakce s rukou, ale naopak je možné řetězit příkazy do složitých sekvencí, které je možné do ruky nahrát nebo si je uložit. Mimo programování, ukládání nebo přímého ovládání ruky umožňuje aplikace také různé uživatelské nastavení, nástroj pro kontrolu kódu před odesláním a pomoc při jeho opravě.

3.2.1. Všeobecný přehled designu aplikace a její funkcí

Po uživatelské stránce tu byla snaha o vytvoření na pohled příjemného a intuitivního pracovního prostředí. Již ikona je stylizována do barev fakulty s logem VUT. Po zapnutí se objeví stylizované informační okno *(IntroWin)*, které se po 7vteřinách přepne na hlavní okno *(MainWin)*. Okno je rozděleno na 3 hlavní oblasti. Nahoře se nachází lišta hlavních nabídek obsahující položky *Soubor, Zobrazení a Nápověda*. V prostřední části je pracovní prostor, který je uživatelem modifikovatelný. Dole je pak stavový řádek, kderý podává uživateli základní informace jako jsou zvolený port, průběh připojování, stav zařízení *(Připojeno/Nepřipojeno)*, aktuálně otevřený projekt, firmware připojené desky nebo řádek kurzoru.



Obr. 13 Ukázka hlavního okna a pomocného okna se schématem

3.2.2. Hlavní lišta Soubor ...

Položka *Soubor* obsahuje klasickou nabídku pro práci se soubory nebo projekty. Tlačítko *Nový*, spojené s klávesovou zkratkou Ctrl+N, založí nový projekt a naplní textový režim povinnou hlavičkou a patičkou. V případě po spuštění aplikace se projekt založí ihned, v případě předchozího založení projektu nebo otevření existujícího program upozornění na ztrátu dat a optá se na další akci.

Tlačítko *Otevřít*, spojené se zkratkou Ctrl+O, zobrazí dialogové okno s třemi filtry pro soubory:

- .robo – soubor zvolen přímo pro projekty aplikace
- .txt – textový soubor, který může také obsahovat kód
- Všechny soubory – možnost pro případ, kdy si uživatel uložil v jiném programu s neobvyklou koncovkou

Koncovka .robo je použita úmyslně, aby nedocházelo k neúmyslnému otevírání a poškozování kódu v programu Poznámkový blok nebo podobném. Také dává možnost filtrovat pouze projekty aplikace i v jinak plném a nepřehledném adresáři s různými soubory. Zabezpečení u otevírání je obdobné jako při zakládání projektu. Po založeném projektu nebo po předchozím otevření souboru je před zobrazením *OpenDialogu* zobrazeno dialogové okno s upozorněním a volbou pokračování a ztrátou dat nebo přerušením a zachováním dat.

Položky *Uložit* a *Uložit jako* mají standardní klávesové zkratky Ctrl+S a Ctrl+Shift+S. Při ukládání jako se vždy objeví dialog uložení. Při volbě uložit aplikace zjišťuje, zda je kód kam uložit. Pokud nemá k dispozici žádné umístnění souboru, otevře *SaveDialog*.

Poslední otevřené zobrazuje soubory, které byly naposled otevřeny a umožňuje rychlé otevření souboru bez otevírání dialogů. Tato položka je schopna pamatovat si až 5 posledních otevření. Adresy umístnění se ukládají do souboru *RecentlyOpen.roboconfig* v adresáři AppData. Při smazání, přejmenování nebo přesunutí souboru a následném pokusu ho otevřít ze seznamu posledních otevřených se otevře dialog s upozorněním, že soubor již není dostupný.

Při testování bylo zjištěno, že aplikace si pamatuje po prvním uložení adresář s projekty. Pro použití v laboratořích bude tato skutečnost praktická, jelikož se jednou založí adresář projektů a pokud ho student nezmění, bude při ukládání a otevírání vždy nabízen právě tento adresář.

Pod menu *Zobrazení* se ukrývají uživatelská nastavení vzhledu, jako jsou fonty, velikosti písma nebo inverze barev. Mimo to je zde i možnost skrýt nebo zobrazit některé panely nástrojů. Textové pole v textovém režimu umožňuje změnu všech zmíněné vlastnosti. Informační panel pouze inverzi barev a vyčištění.

Menu *Nápověda* nabízí zobrazení detailů *O programu* a poté se tu nachází pomůcka pro rozlišení jednotlivých serv *Rozvržení serv.* Po kliknutí se zobrazí nezávislé okno bez možnosti změny velikosti. Přes celou plochu je na pozadí nastaven obrázek jednoduchého schématu robotické ruky. Opticky je rozdělena podle serv na barevné oblasti. Barvy odpovídají barvám bloků v grafickém režimu. Pro použití i v textovém režimu jsou zde uvedený i názvy jednotlivých serv. Při kliknutí pravým tlačítkem kamkoliv na plochu okna je možní zvolit, zda bude okno dokované vždy nad všemi ostatními okny nebo zda se bude podřizovat ostatním oknům.

3.2.3. Pracovní prostor

Pracovní prostor je rozdělen mnoha komponentami na několik menších celků. Tato část se věnuje jen hlavnímu rozdělení.

První rozdělení je tvořeno prvkem *SplitContainer*, který rozděluje prostor na dva a umožňuje jejich velikost měnit posuvem. Dolní část obsahuje pouze informační panel, kde se vypisují hlášení o provedených akcích nebo chybách. Informační panel je tvořen *ListBoxem* s nabídkou po kliknutí pravým tlačítkem myši. Ta obsahuje možnost vyčištění všech hlášení a inverze barev. Horní část dále obsahuje *ToolStripContainer.* Ten umožňuje umístit do něj přesouvatelné nástrojové panely po obvodu a uprostřed je použitelný prostor.[12] Ten je obsazen dalším formátovacím prvek, *TabControl*, který umožňuje přepínat mezi různými obsahy a vždy zobrazuje jen jeden panel. Použitý *TabControl* obsahuje 3 panely, které obsahují jednotlivé režimy ovládání ruky.

3.2.4. Panely nástrojů

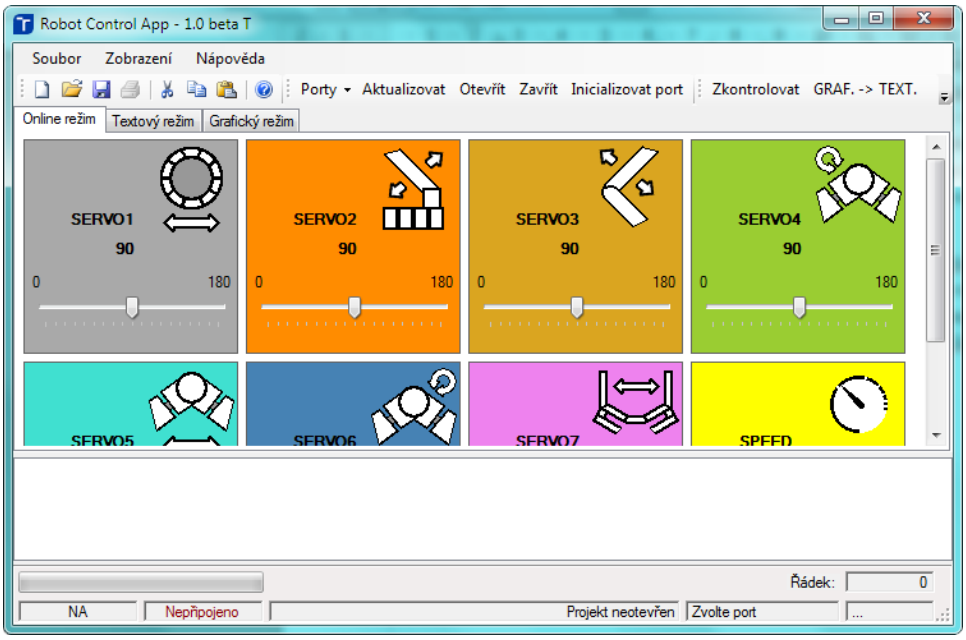
Panely nástrojů v *ToolStripContraineru* je možné přesouvat nejen v prostoru horní hrany, ale je možné je přemístit i do spodní nebo bočních částí. Při kliknutí na prostor panelů pravým tlačítkem se otevře nabídka pro zobrazení nebo schování panelů.

Hlavní panel, nástrojů se základními funkcemi jako je práce se soubory nebo textem, není možné schovat. Panel *Připojení* obsahuje základní ovládací funkce pro práci se sériovým portem. Jelikož je používáno pouze Arduino, není potřeba měnit baudrate nebo jiné parametry kromě čísla portu. Panel *Kompilace* umožňuje kontrolovat a kompilovat kód. Při kontrole je zároveň i posílám do řízení. Při práci v grafickém režimu je nutné převést před kompilací kód, proto je tato funkce převodu také v panelu kompilace. Panel *Serial příkazy* je po spuštění aplikace defaultně skryt. Obsahuje textové pole pro přímé psaní a posílání příkazů a tlačítko pro odeslání. Jedná se o již zmíněnou druhou možnost online ovládání ruky.

3.2.5. Online režim

Online režim je prvním režimem z karet prvku *TabControl*. Panel obsahuje jediný vložený *FloatLayoutPanel*, který je roztažen na celou plochu a tvoří pracovní prostor. Pro jednotlivé ovládací prvky byl vytvořen objekt *UserControl*. Jedná se o unikátní prvek, který si programátor vytvoří dle vlastní potřeby. Všechny použité „uživatelské“ objekty v aplikaci mají podobný vzhled. Jsou to čtverce o rozměrech 170x170px (Online režim) nebo 150x150px (Grafický režim). Obsahují *Label* (needitovatelný prostý text) s názvem bloku, jednu ikonu o rozměru 70x70px, která funkci bloku vyjadřuje a ovládací prvek, který nastavuje prvku určitou hodnotu. Pracovní plocha obsahuje celkem 8 panelů, 7 pro serva a jeden pro změnu rychlosti. Panely v online režimu využívají *TrakBar*, prvek se šoupátkem pro nastavení číselné hodnoty.

Při uchopení šoupátka a posuvem se hodnota šoupátka ukazuje na labelu daného panelu. Po puštění se do řízení, pokud je připojeno, pošle zjednodušený příkaz pro nastavení polohy nebo rychlosti. Tvar zjednodušeného příkazu je PRIKAZHODNOTA, zapisuje se bez mezer, bez závorek a bez středníku. Totožné příkazy se používají v panelu nástrojů *Serial příkazy.*



Obr. 14 Online režim aplikace

3.2.6. Textový režim

Panel textového režimu obsahuje jeden velký multiřádkový *TextBox*, který vyplňuje celý vnitřní prostor panelu. Uživatel má možnost změnit si font písma, velikost nebo invertovat barvy po otevření nabídky pravým tlačítkem. S tímto textovým polem jsou spojeny i nástroje kopírování a vložení na panelu nástrojů.

Textový režim je datovým uzlem pro práci se soubory i posílání dat do Arduina. Do souborů se ukládá obsah právě tohoto textového pole a stejně tak se do něj vypisuje obsah otevřených souborů. Při spuštění kontroly se kontroluje kód uvnitř pole a z tohoto kódu se při kompilaci vytváří programovací řetězec, který je posílán do mikrokontroléru. I data z grafického režimu jsou generována do podoby jazyku z textového režimu a jsou zde k náhledu. Obsah pole náhledu *(ListBox)* je poté nakopírován do pole textového režimu.

Po spuštění aplikace je pole textového režimu zcela prázdné. Uživatel si může zvolit, zda začne psát kód hlavičky a patičky souboru nebo si nechá povinnou hlavičku a patičku vygenerovat tlačítkem *Nový*. Hlavička je zde pro rozpoznání souboru projektu. Při otevírání souboru je možné otevřít kromě souboru .ROBO také jakýkoliv textový soubor .TXT. Uživateli to umožňuje napsat si kód bez programu a program využít pro kontrolu a nahrání do řízení. Ovšem to dává prostor pro chyby a kompilaci nevhodných souborů. Patička slouží jako bezpečná zarážka pro debugger a kompilaci kódu. Jak debugger, tak kompilátor využívají řádek s příkazem *end;* jako ukončení činnosti.

Níže je ukázka kódu nového projektu pouze s hlavičkou a patičkou:

```
begin ROBO;
//Zde pis prikazy ukoncene strednikem
//Seznam prikazu naleznes v napovede
end;
```

Příkazy popisující činnost robotické ruky jsou také striktně dané. Kromě použití všech velkých písmen a správných slov, je povinné použití závorek, hodnota v závorkách a ukončení příkazu středníkem. I když je možné kompilovat kód bez středníků, byla tato funkce zakomponována jako tréning uživatelů pro pevně danou syntaxi. Chybějící středník je při programování velmi častá chyba a ani kompilátory většiny profesionálních vývojových systémů tuto chybu neodpustí.

Ovládací příkazy ruky se dají rozdělit na příkazy, které vykonávají viditelnou činnost a příkazy, jejichž provedení se projeví až v dalším příkazu nebo nečinností. Příkazy pohybové mají všechny v názvu SERVO a poté číslo serva. Číselný parametr těchto příkazů nabývá hodnot možných poloh daného serva (0°-180°). Příkaz SPEED_ nastaví při svém provedení hodnotu časové prodlevy mezi změnou polohy serva o 1°, takže se projeví až při dalším příkazu SERVO. Příkaz DELAY_ pouze naplní hodnotu funkce časové prodlevy v Arduinu a tím způsobí nečinnost ruky po zvolenou dobu.

Všechny příkazy mají omezený rozsah i krok, například SERVO má krok 5 a rozsah 0 – 180. Příkaz SPEED_ je potřeba při určité fázi invertovat, jelikož v kódu se větším číslem nastavuje větší rychlost, avšak prakticky znamená větší prodleva mezi posunem o krok znamená nižší rychlost. Příkaz DELAY_ má krok 50 ms a rozsah 0 – 1000 ms.

Kromě standardních příkazů je možné použít dvou lomítek pro komentář. Takový komentář končí s novým řádkem. Pro zmenšení počtu řádků kódu a tím čas kontroly je prázdný řádek brán jako chyba, aby nebylo v kódu příliš zalomení. V případě nutnosti zalomení je možné ho „zakomentovat“. Kód by pak vypadal následovně:

```
begin ROBO;
//
//
SERVO1(30);
//
//
end;
```

Ani takovýto kód by neměl v rozsahu několika jednotek až desítek dělat problém. Při delších sekvencích by však již mohla narůstat délka kontroly s narůstajícím balastem (nefunkční zakomentovaný kód).

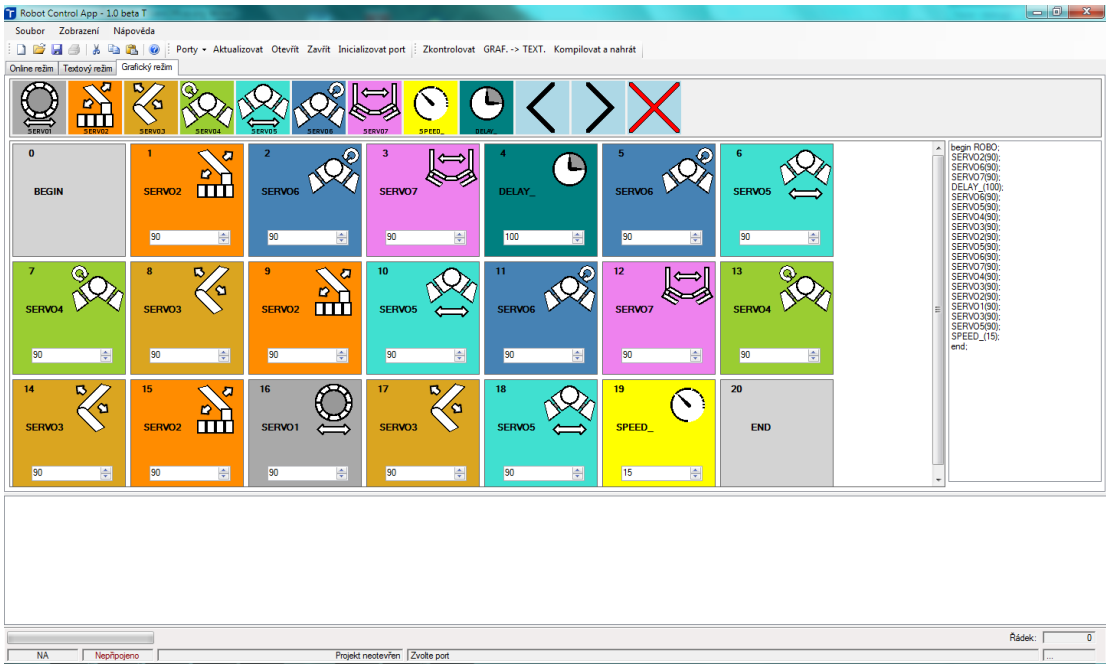
3.2.7. Grafický režim

Grafický režim je rychlým a jednoduchým způsobem naprogramování pohybu. Panel režimu obsahuje nahoře nástrojovou lištu (při malé velikosti jsou některé funkce rolovatelné), která je obalena prvkem *FlowLayoutPanel* a obsahuje bloky reprezentující všechny existující příkazy programu a tři ovládací prvky. Všechny bloky mají stejné barvy odpovídající dané funkci, stejně jako je tomu v online režimu. Po krátkém odzkoušení ovládání v online režimu může uživatel bez větších problémů graficky naprogramovat první sekvenci. Hned pod panelem je další *FlowLayoutPanel* fungující jako pracovní plocha. Již od spuštění aplikace obsahuje dva neodstranitelné koncové prvky, begin a end, reprezentující hlavičku a patičku souboru. Vpravo je k nahlédnutí automaticky generovaný kód. Tento kód má dvě odlišné funkce. Technickým důvodem je výhodnější práce při přesouvání a mazání řádků v *ListBoxu*. Uživatelským důvodem je asociace grafického programování s textovým a pomoc textové programování lépe pochopit.

Programování v tomto režimu je velmi jednoduché. Stačí uchopit a přetáhnout požadovaný příkaz na pracovní plochu. Správné uchopení je možné zkontrolovat podle kurzoru myši, který se změní. Plocha zase znázorní změnou barvy na tmavě šedou přítomnost kurzoru nad ní. Při uvolnění objektu nad plochou nebo odchodem z plochy se vrátí barva pozadí zpět na světlou. Přetažení objektu na plochu navíc přidá přetahovaný objekt na poslední místo před blokem end.

Na pracovní ploše se přidávají objekty vždy nakonec a podobně jako při psaní, zleva doprava a shora dolů. Každý přidaný blok má barvu pozadí jako přetahovaný objekt, obsahuje číslo s pořadím v řetězci (blok begin má pořadí 0), název bloku podle příkazu, ikonu příkazu a prvek *NumericUpDown*, kterým se nastavuje hodnota. Nastavením hodnoty na bloku se zároveň mění hodnota v náhledu. Blok reprezentující určitý příkaz nelze změnit na prvek reprezentující jiný příkaz. Rozdílné barvy bloků jsou zvoleny úmyslně, aby uživateli pomohli je jednoduše rozlišit jak v panelu nástrojů, tak na pracovní ploše. Mimo to jsou stejné barvy použity i v *Rozvržení kloubů* pod položkou *Nápověda*.

Mimo přetahovatelných bloků jsou tu klikací nástroje na úpravu řetězce. Jsou to: přesun vlevo, vpravo a vymazání bloku. V neaktivním stavu mají bloky světle modrou barvu. Při kliknutí se barva změní na světle červenou a při přejíždění po pracovní ploše se zobrazí kurzor znázorňující danou operaci. Při kliknutí na určitý panel se panel přesune vlevo, vpravo a nebo se odstraní. Při všech těchto úkonech se změní i náhled kódu a indexy okolních bloků dle nového pořadí. Aktualizace v náhledu je velice důležitá, aby se poté správná data přenesla do textového režimu.



Obr. 15 Grafický režim s poskládanou sekvencí z bloků

4. Závěr

V práci je popsán princip funkce robotické ruky včetně parametrů, které rozhodli o volbě realizované konstrukce. Důkladná znalost pomohla v návrhu možných technických řešení a určení té nejvhodnější, konkrétně pak technologie 3D tisku a pohon realizován servy a řízen Arduinem s DPS řízení pro zapojení a ovládání bez PC. V rámci práce byla robotická ruka sestrojena a osazena servy. Dále byl vytvořen program pro mikrokontrolér, který je schopný komunikovat s počítačem a ovládat ruku. Pro lepší zapojení serv a přidání ovládacích prvků byla navržena a vyrobena DPS, která sedí přímo do desky mikrokontroléru. Také byla vytvořena aplikace pro komunikaci a ovládání Arduina a ruky. Kromě řízení v reálném čase umožňuje programování pohybových sekvencí. Aplikace podporuje programování textové a především grafické pomocí grafických bloků.

S vytvořeným řešením je možné ovládat a programovat robotickou ruku a použít ho v laboratořích jako učební pomůcku pro rozvíjení schopností studentů.

5. Seznam použitých zdrojů

[1] SOBOTA, D. *Návrh řízení robotické ruky.* Brno: Vysoké učení tehnické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 18 s.Vedoucí semestrální práce Ing. Petr Vyroubal.

[2] Upraveno z WWW:
http://www.thingiverse.com/thing:30163/#files

[3] Upraveno z WWW:
http://www.rc-auta.eu/images/servo_schema.gif

[4] *Serva* [cit. 2016-05-08]. Dostupné z WWW:
http://vlastikd.webz.cz/bastl/serva.htm

[5] Převzato z WWW:
http://vlastikd.webz.cz/bastl/images/servo3.gif

[6] Převzato z WWW:
http://hitecrcd.com/images/products/full/128_1_HS-311_Standard_Economy_Servo-1.jpg

[7] Převzato z WWW:
http://www.pelikandaniel.com/products/1MA009/b_0.jpg

[8] *ArduinoBoardMega2560* [cit. 2015-11-30]. Dostupné z WWW:
https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[9] *Universal Serial Bus* [cit. 2015-12-07]. Dostupné z WWW:
https://cs.wikipedia.org/wiki/Universal_Serial_Bus

[10] Převzato z WWW:
https://ardubasic.files.wordpress.com/2014/04/arduino_mega_rev_3_bb.png

[11] LIBERTY, Jesse. *Naučte se C++ za 21 dní. 2., aktualiz. vyd.* Překlad Josef Pojsl, Karel Voráček. Brno: Computer Press, 2007.

[12] KRUGLINSKI, David, Scot WINGO a George SHEPHERD. *Programujeme v Microsoft Visual C++*. Praha: Computer Press, 2000.

6. Seznam obrázků